Soluzioni della Prova Scritta di Sistemi di Elaborazione Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

30 gennaio 2009

1. (a) Il programma prende in ingresso un valore n da riga di comando e stampa a video tutte le coppie di numeri (x, y) che soddisfano le seguenti condizioni: i) x e y sono minori o uguali a n; ii) y è maggiore di x; iii) la somma dei divisori propri di x è uguale a y e viceversa. (x e y sono *numeri amicabili*).

   (b) Una possibile traduzione è la seguente:

```
        .text
        .global f2
f2:     pushl %ebp
        movl  %esp, %ebp
        subl  $12, %esp
        pushl %ebx
        pushl %edi
        pushl %esi
        #-4(%ebp) e' i
        #-8(%ebp) e' j
        #-12(%ebp) e' a
        movl  $0, -12(%ebp)
        movl  $2, -4(%ebp)
        movl  12(%ebp), %ebx
for1:   movl  -4(%ebp), %edi
        cmpl  %edi, 8(%ebp)
        jl    finefor1
        movl  %edi, -8(%ebp)
        incl  -8(%ebp)
for2:   movl  -8(%ebp), %esi
        cmpl  %esi, 8(%ebp)
        jl    finefor2
        pushl -4(%ebp)
        call  f1
        addl  $4, %esp
        cmpl  %eax, -8(%ebp)
        jne   avanti
        pushl -8(%ebp)
        call  f1
        addl  $4, %esp
        cmpl  %eax, -4(%ebp)
        jne   avanti
prova:  movl  -12(%ebp), %eax
        movl  -4(%ebp), %edi
        movl  %edi, (%ebx, %eax, 4)
        incl  -12(%ebp)
        movl  -12(%ebp), %eax
        movl  -8(%ebp), %esi
        movl  %esi, (%ebx, %eax, 4)
        incl  -12(%ebp)
avanti: incl  -8(%ebp)
        jmp   for2
finefor2:
        incl  -4(%ebp)
        jmp   for1
finefor1:
        movl  -12(%ebp), %eax
        popl  %esi
        popl  %edi
        popl  %ebx
        leave
        ret
```

2. (a)
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAXSTR 100
#define MAXLINE 1024

int main(int argc, char* argv[])
{
```

```
        char buf[MAXLINE];
        char pat[MAXSTR][MAXLINE];
        FILE *f;
        int n, i, j;

        if (argc < 3) {
                fprintf(stderr, "Uso: %s <file>|- <file> ...\n", argv[0]);
                exit(1);
        }

        f = stdin;
        if (strcmp(argv[1], "-") != 0) {
                if ( ! (f = fopen(argv[1], "r")) ) {
                        perror(argv[1]);
                        exit(1);
                }
        }

        n = 0;
        while (n < MAXSTR && fgets(pat[n], MAXLINE, f)) {
                int l = strlen(pat[n]);
                pat[n][l - 1] = '\0';
                n++;
        }
        fclose(f);

        for (i = 2; i < argc; i++) {
                if ( ! (f = fopen(argv[i], "r")) ) {
                        perror(argv[i]);
                        continue;
                }
                while ( fgets(buf, MAXLINE, f) )
                        for (j = 0; j < n; j++)
                                if (strstr(buf, pat[j]))
                                        printf("%s: %s", argv[i], buf);
                fclose(f);
        }
        return 0;
}
```

(b)
```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>

#define MAXNAME 255
#define MAXFILES 100

int main(int argc, char* argv[])
{
```

```
struct stat st;
DIR *d;
struct dirent *e;
char *c_argv[MAXFILES + 3];
char buf[MAXFILES][MAXNAME];
int i, n, fd[2], l;

if (argc != 2) {
        fprintf(stderr, "Uso: %s <dir>\n", argv[0]);
        exit(1);
}

if ( chdir(argv[1]) < 0 ) {
        perror(argv[1]);
        exit(1);
}

if ( !(d = opendir(".")) ) {
        perror(argv[1]);
        exit(1);
}

c_argv[0] = "cerca";
c_argv[1] = "-";
n = 0;
while ( n < MAXFILES && (e = readdir(d)) ) {
        if ( stat(e->d_name, &st) < 0 ) {
                perror(e->d_name);
                continue;
        }
        if (!S_ISREG(st.st_mode))
                continue;
        strncpy(buf[n], e->d_name, MAXNAME);
        buf[n][MAXNAME - 1] = '\0';
        c_argv[n + 2] = buf[n];
        n++;
}
c_argv[n + 2] = NULL;
closedir(d);

if ( pipe(fd) < 0) {
        perror(argv[0]);
        exit(1);
}

switch ( fork() ) {
case -1:
        perror(argv[0]);
        exit(1);
case 0:
        close(0);
        dup(fd[0]);
        close(fd[0]);
```

```
                close(fd[1]);
                execvp("cerca", c_argv);
                perror("cerca");
                exit(1);
        default:
                break;
        }

        switch ( fork() ) {
        case -1:
                perror(argv[0]);
                exit(1);
        case 0:
                close(1);
                dup(fd[1]);
                close(fd[0]);
                close(fd[1]);

                for (i = 0; i < n; i++)
                        printf("%s\n", buf[i]);

                exit(0);
        default:
                break;
        }

        close(fd[0]);
        close(fd[1]);
        wait(0);
        wait(0);
        return 0;
}
```